View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by DBIS EPub
In: Proc. of WETICE 2007 International Workshops, Agile Cooperative Process-Aware Information Systems (ProGility), pp. 376-378. IEEE Computer Society Press.

Data-driven Design of Engineering Processes with COREPRO$_{Modeler}$*

Dominic Müller, Manfred Reichert
Information Systems Group,
University of Twente, The Netherlands
{d.mueller, m.u.reichert}@utwente.nl

Joachim Herbst, Florian Poppa
Dept. GR/EPD, DaimlerChrysler AG
Group Research & Advanced Engineering, Germany
{joachim.j.herbst,florian.poppa}@daimlerchrysler.com

Abstract

Enterprises increasingly demand IT support for the coordination of their engineering processes, which often consist of hundreds up to thousands of sub-processes. From a technical viewpoint, these sub-processes have to be concurrently executed and synchronized considering numerous interdependencies. So far, this coordination has mainly been accomplished manually, which has resulted in errors and inconsistencies. In order to deal with this problem, we have to better understand the interdependencies between the sub-processes to be coordinated. In particular, we can benefit from the fact that sub-processes are often correlated to the assembly of a product (represented by a product data structure). This information can be utilized for the modeling and execution of so-called data-driven process structures. In this paper, we present the COREPRO demonstrator that supports the data-driven modeling of these process structures. The approach explicitly establishes a close linkage between product data structures and engineering processes.

1. Introduction

Complex engineering processes, such as the development of a car, consist of many interdependent sub-processes. When realizing IT support for the coordination of these sub-processes, the challenge is to explicitly define the dependencies between them. Case studies, we conducted in the automotive industry, have shown that the coordination of sub-processes is usually based on the assembly of the product to be developed [2, 7]. According to [10], we use the notion *data-driven process structures* in this context (cf. Fig. 1). We identified numerous scenarios for such data-driven process structures. One example is the verification process for the electrical system of a car. This system consists of about 270 interconnected components [4]. To verify the functionality of the system, for each component several sub-processes (e.g., dealing with *test preparation*, *simulation test*, and *test drive*) have to be executed and synchronized. The synchronization behavior is determined by sub-process dependencies and their relationships to product components, such as "*prepare all engine components before starting the simulation test for the engine*". As a consequence, we obtain a close connection between data and processes as illustrated in Fig. 1. The introduction of complex car features (e.g., driving assistants that are controlling the engine or the brake system) will further increase the number of component relations. They lead to additional sub-process synchronizations necessary to verify the functionality of these features. Altogether, a data-driven sub-process structure may comprise of thousands of sub-processes and sub-process dependencies in the presented scenario. Hence, the coordination will become more and more complex and very error-prone, if done manually (which is the normal case in current development projects).

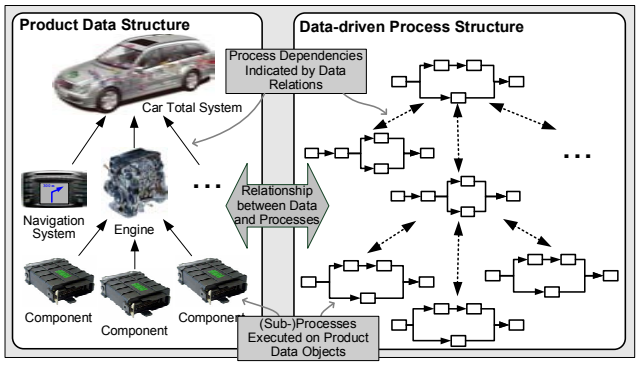

Figure 1. Data-driven Process Structure

IT support for the modeling of data-driven process structures must meet four requirements. First, it must enable the description of the (product) data structure, i.e., its objects and their relations. Second, a concept for associating sub-processes with each (product data) object has to be defined. Third, the definition of dependencies between sub-processes for different objects has to be enabled. In particular, object relations have to be associated with several sub-

*This work has been funded by *DaimlerChrysler AG Group Research*

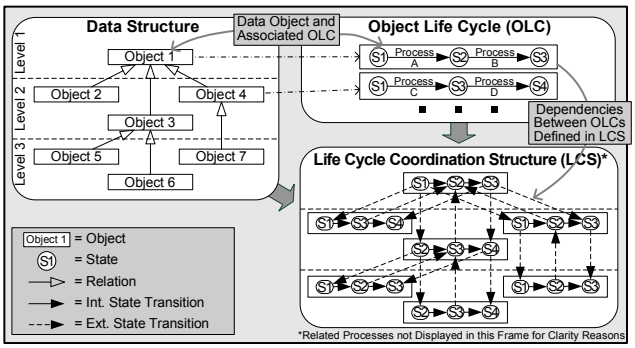

Figure 2. Overview of our Approach

process dependencies (i.e., a relation may lead to several sub-process synchronizations). Fourth, it must be possible to generate an activity-centered process structure based on the specified information.

Current approaches in literature and practice do not meet these requirements. Neither activity-centered workflow systems (e.g., *Production Workflow* [5]) nor data-centered approaches (e.g., *Case Handling* [1]) enable the independent association of several sub-processes to objects and the utilization of object relations for defining sub-process dependencies. Approaches utilizing product structures for deriving process structures (e.g., *Product Driven Workflow Design* [9]) do not allow for the detailed specification of multiple sub-process dependencies based on an object relation. Even approaches aiming at the support of engineering processes (e.g., *AHEAD* [3]) lack mechanisms for mapping object relations to sub-process dependencies. Currently, the way to cope with these problems is the manual integration of all necessary information into one large and inflexible process (structure). However, that generates high efforts for its definition, monitoring, and particularly for its maintenance, e.g., when changing the data structure [8].

In this paper, we present the *COREPRO* demonstrator, which enables the modeling of (product) data-driven process structures. Related concepts are discussed in Section 2. Section 3 gives a short description of our demo.

2. COREPRO Features

The *COREPRO* project is developing solutions for the design, execution, and flexible adaptation of data-driven process structures. The design goals are separated definition of the different information for data structures and sub-processes, and creation of the requested data-driven process from the given information. To realize this, *COREPRO* follows the *Model Driven Architecture* approach (MDA) [6]. In particular, it enables the definition of data structures as well as their relationships to engineering sub-processes.

In COREPRO, sub-processes can be associated with (data) objects by defining *Object Life Cycles* (OLC). While the data structure describes the relations between objects (cf. Fig. 2), an OLC specifies possible states of a single (product data) object during its lifetime (cf. Fig. 2). The OLC is a transition system with data states and *internal state transitions*. Every state transition has an associated (sub-)process. A state transition is activated by executing its associated (sub-)processes, which is modifying the object (cf. Fig. 2). A car component, for example, has several states (e.g., `Ready` followed by `Prepared`) which are changed by executing processes for this component (e.g., the `Test Preparation` sub-process triggers a state transition from `Ready` to `Prepared`). Dependent on the sub-process result, alternative states may be activated. Altogether, the OLC defines the association of sub-processes to a specific object.

However, associating objects and processes by defining an OLC for every object is only one part of our solution. We also have to deal with dependencies between sub-processes associated to different objects. An example for this was given in Section 1: every engine component must have reached state `Prepared` before starting the `Simulation Test` sub-process for the whole engine. These dependencies can be defined in the *Life Cycle Coordination Structure* (LCS). Therefore, the LCS contains an OLC for every object and allows for the definition of *external state transitions*, which connect states of the included OLCs. In the LCS in Fig. 2, for example, the states `S1` of the OLCs for the related objects `1` and `2` are connected. Altogether, the LCS describes possible states of the whole data structure during its lifetime. While we avoid concurrently activated states (and thus concurrently executed sub-processes) for single OLCs, there are concurrently activated states in the LCS (one per OLC) during runtime.

The LCS constitutes a platform independent model of the data-driven process. Based on transformation rules activity-centered data-driven process structure can be generated, e.g., using *Business Process Modeling Notation* (BPMN) or *Business Process Execution Language* (BPEL).

3. Demo Description

We implemented our proof-of-concept demonstrator using the *Eclipse Modeling Framework* (EMF). $COREPRO_{Modeler}$ supports the graphical modeling of a data-driven process according to the concepts presented in Section 2. It comprises editors for the data structure, the object life cycle (OLC), and the life cycle coordination structure (LCS). Additionally, a generator tool for the activity centered representation of the data-driven process structure is realized.

For demonstration purposes, we have modeled the example presented in Fig. 2, where a hierarchical data structure

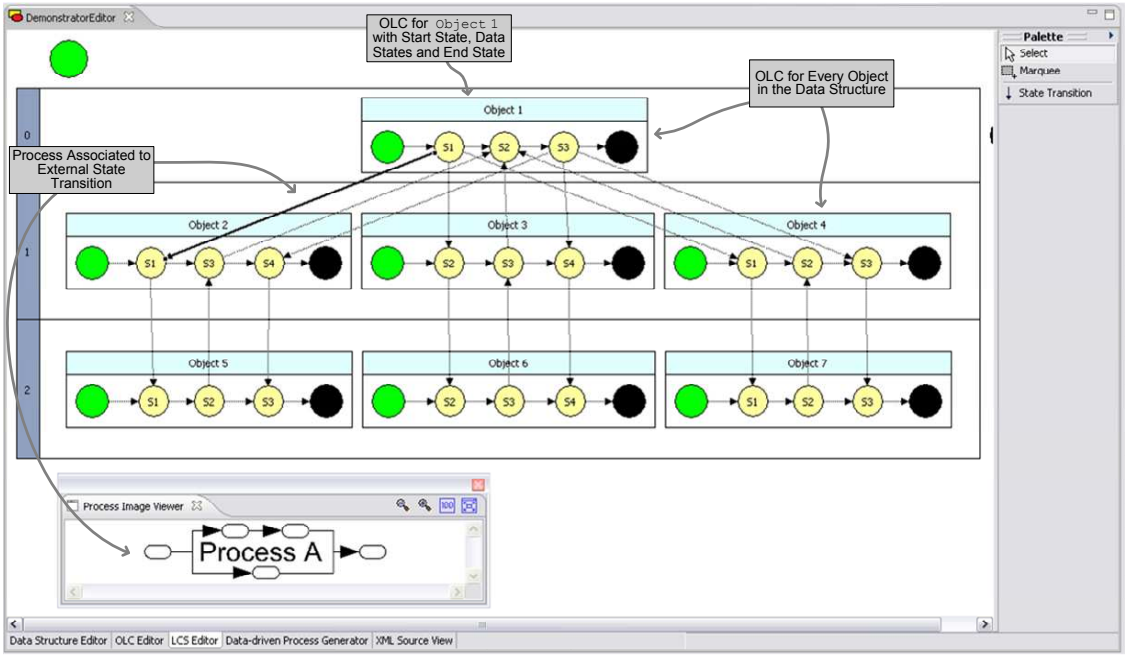

Figure 3. Definition of the Life Cycle Coordination Structure

with ten objects (arranged in three levels) is shown. Every object has an associated OLC with three data states.

When modeling data-driven processes, we first have to define the data structure. For this purpose, labeled objects have to be created and interconnected with the *data structure editor*. Afterwards, an OLC can be specified for every object in order to associate the sub-processes. Therefore, every state transition can be linked with a premodeled subprocess in the *OLC Editor*.

After having modeled an OLC for each object, the user specifies the runtime behavior for the whole data structure. The *LCS Editor* (cf. Fig. 3) provides the modeled OLC for every object. To express the relations between them, the OLCs can be interconnected with external state transitions. They can also be associated with sub-processes realizing the synchronization (e.g., `Process A` in Fig. 3).

The activity-centered representation of the data-driven process structure is then derived by mapping the elements of the LCS to control flow constructs.

The demonstrator enables the modeling of data-driven process structures by defining data structures and subprocess dependencies according to object relations. The connection between data and processes can be specified in a life cycle layer defining the runtime behavior for the whole data structure. By transforming the specified models, COREPRO enables the generation of a platform dependent process structure. The presented demonstrator is used for a first proof-of-concept case study in industry, where it enables the product oriented modeling of car development processes. We further plan to implement an instantiation mechanism, which reduces modeling efforts. In particular, we support defining OLC templates and external state transitions on model level for object types and relation types.

References

[1] W. Aalst, M. Weske, and D. Grünbauer. Case handling: A new paradigm for business process support. *DKE*, 53(2), 2005.

[2] U. Bestfleisch, J. Herbst, and M. Reichert. Requirememts for the workflow-based support of release management processes in the automotive sector. In *ECEC*, 2005.

[3] D. Jäger, A. Schleicher, and B. Westfechtel. AHEAD: A graph-based system for modeling and managing development processes. In *AGTIVE*, LNCS 1779, 1999.

[4] E. Knippel and A. Schulz. Lessons learned from implementing configuration management within electrical/eletronic development of an automotive OEM. In *INCOSE '04*, 2004.

[5] F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, 2000.

[6] J. Miller and J. Mukerji. OMG MDA Guide, 2003.

[7] D. Müller, J. Herbst, M. Hammori, and M. Reichert. IT support for release management processes in the automotive industry. In *BPM'06*, LNCS 4102, 2006.

[8] D. Müller, M. Reichert, and J. Herbst. Enabling flexibility of data-driven process structures. In *DPM'06*, LNCS 4103, 2006.

[9] H. Reijers, S. Limam, and W. Aalst. Product-based workflow design. *MIS*, 20(1), 2003.

[10] S. Rinderle and M. Reichert. Data-driven process control and exception handling in process management systems. In *CAiSE'06*, LNCS 4001, 2006.